Problem 1

A neural network uses a hierarchical feature representation of the input space and adaptive basis functions to linearly separate the input space. The basis functions are adaptive in the sense that their coefficients are not hand-crafted but determined by learning.

Problem 2

$$\begin{aligned}
\tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\
&= \frac{e^x}{e^x + e^{-x}} - \frac{e^{-x}}{e^x + e^{-x}} \\
&= \frac{1}{1 + e^{-2x}} - \frac{1}{1 + e^{2x}} \\
&= \sigma(2x) - \sigma(-2x) \\
&= \sigma(2x) - (1 - \sigma(2x)) \\
&= \sigma(2x) - 1 + \sigma(2x) \\
&= 2\sigma(2x) - 1
\end{aligned}$$

It immediately follows that

$$\sigma(x) = \frac{\tanh(\frac{x}{2}) + 1}{2} \tag{2.1}$$

By (2.1), if we take all the *input* hidden-layer weights σ multiply them by $\frac{1}{2}$ and scale the all *output* hidden-layer weights by $\frac{1}{2}$ and add $\frac{1}{2}$ to their bias we get the exact same neural network function output using tanh activation functions for the hidden layer.

Problem 3

$$\begin{aligned}
\frac{\partial}{\partial x}(\tanh(x)) &= \frac{\partial}{\partial x}(\frac{e^x - e^{-x}}{e^x + e^{-x}}) \\
&= \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2} \\
&= 1 - \tanh^2(x)
\end{aligned}$$

It simplifies the mathematical derivations and the implementation, specifically the computation of the local gradient at an intermediate node and the backpropagation of the gradient to the lower layers.

Problem 4

For deriving the new loss function, we follow a similar approach as in cf. Lecture Linear Classification (slide 28). We use $\tanh(\cdot)$ instead of $\sigma(\cdot)$. To interpret Ber $\sim p(y|x)$ as probability we rescale it to be in $[0, 1]$.

$$p(y=1|x) = \frac{\tanh(x)+1}{2}, \quad y=1$$
$$p(y=-1|x) = 1 - \frac{\tanh(x)+1}{2}, \quad y=-1$$

or more compactly as: $x^{\frac{1+y}{2}} \cdot (1-x)^{\frac{1-y}{2}}$

$$\begin{aligned} E(W) &= -\sum_{i=1}^{N} \frac{1+y}{2}\log(\frac{\tanh(x_i,W)+1}{2}) + \frac{1-y}{2}\log(1-\frac{\tanh(x_i,W)+1}{2}) \\ &= -\sum_{i=1}^{N} \frac{1+y}{2}\log(\frac{f(x_i,W)+1}{2}) + \frac{1-y}{2}\log(1-\frac{f(x_i,W)+1}{2}) \end{aligned}$$

Clearly the loss is zero for correctly classified datapoints and non-negative for missclassified datapoints. Furthermore an output of $\tanh(x), x \to -1$ and $\tanh(x), x \to 1$ results in low respectively high probability value.
The activation function to be chosen is $\tanh(\cdot)$

Problem 5

Let $z_i = y_i - wx_i$

$$\frac{\nabla_{E(w)}}{\partial w} = \sum_{i=1}^{N} \frac{\partial}{\partial z_i}E(w) \cdot \frac{\partial}{\partial w}z_i \tag{5.1}$$

$$\frac{\partial}{\partial z_i}E(w) = \begin{cases} y_i - wx_i, \text{ if } |z| < 1 \\ 1, \text{ otherwise} \end{cases} \tag{5.2}$$

$$\frac{\partial}{\partial w}z_i = -x_i \tag{5.3}$$

Plugging (5.2), (5.3) in (5.1) we have

$$\frac{\nabla_{E(w)}}{\partial w} = \begin{cases} -\sum_{i=1}^{N}(y_i - wx_i)x_i + \lambda w, \text{ if } |z| < 1 \\ -\sum_{i=1}^{N} x_i + \lambda w, \text{ otherwise} \end{cases}$$

Problem 6

We would stop training approximately at iteration 50, because while the training error keeps decreasing we have an increase in the validation error, which is a sign of overfitting. After we take the weights of the network at this iteration we test it one final time on the test set. Note that we do not get the optimal test error as we would if we would continue with the training until iteration 80.

Problem 7

$$
\begin{aligned}
y &= \log(\sum_{i=1}^{N} e^{x_i}) \\
&= \log(\sum_{i=1}^{N} \frac{e^a}{e^a} e^{x_i}) \\
&= \log(e^a \sum_{i=1}^{N} e^{-a} e^{x_i}) \\
&= \log(e^a) + \log(\sum_{i=1}^{N} e^{-a} e^{x_i}) \\
&= a + \log(\sum_{i=1}^{N} e^{x_i - a})
\end{aligned}
$$

Problem 8

$$
\begin{aligned}
\frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}} &= \frac{e^a}{e^a} \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}} \\
&= \frac{e^{x_i - a}}{\sum_{i=1}^{N} e^{x_i - a}}
\end{aligned}
$$

Problem 9

The binary cross-entropy is equivalent to

$$
\begin{aligned}
-[y\log(\sigma(x)) + (1-y)\log(1-\sigma(x))] &= -[y\log(\frac{e^x}{1+e^x}) + (1-y)\log(1 - \frac{e^x}{1+e^x})] \\
&= -[y\log e^x - y\log(1+e^x) + (1-y)(-\log(1+e^x))] \\
&= -[yx - \log(1+e^x)] \\
&= -yx + \log(1+e^x) \\
&= -yx + \log(e^x(e^{-x}+1)) \\
&= \log(e^x) - yx + \log(e^{-x}+1) \\
&= x - yx + \log(1+e^{-x}) \\
&= \max(x,0) - yx + \log(1+e^{-|x|}) \qquad \text{(see below)}
\end{aligned}
$$

The binary cross-entropy and in general any well-defined loss function must satisfy the following properties : For correctly classified datapoints the loss is zero and for incorrectly classified points it is non-negative. This can be easily verified for the binary cross-entropy, e.g. if x is a large positive score and y = 0, i.e. x is missclassified then we get a non-negative contribution to the loss, conversely if y = 1 then the loss is zero. The stable binary cross-entropy must also satisfy this properties.

For $x \to -\infty$ we get an overflow when implementing it on a computing machine. To overcome this issue, we can map any negative x to a positive value using the abs($\cdot$) function. To ensure that missclassified datapoints contribute with a non-negative value to the loss we have to add the term $\max(x, 0)$ (In case of correctly classified datapoints this term either is zero or cancels out with yx) to the loss function.